**ROMÂNIA**

**MINISTERUL APĂRĂRII NAȚIONALE**

**ACADEMIA TEHNICĂ MILITARĂ „FERDINAND I”**

**Facultatea de Sisteme Informatice și Securitate Cibernetică**

**Departamentul de Calculatoare și Securitate Cibernetică**



***Utilizare senzor ANALOGIC DE ROTAȚIE***

***Platforma de dezvoltare frdm-kl25z***

Std. sg. maj. Bureacă Angela-Emilia

Std. sg. maj. Huțanu David

Grupa C114C

**București**

**2023**

Cuprins

[1. Prezentarea senzorului DFR0054 3](#_Toc93045592)

[2. Scop proiect 4](#_Toc93045593)

[3. Conectare senzor – placă de dezvoltare 5](#_Toc93045594)

[4. Descriere program 6](#_Toc93045595)

[4.1. Funcția main 6](#_Toc93045596)

[4.2. Inițializarea modulelor 6](#_Toc93045597)

[4.2.1. Inițializarea modulului UART 7](#_Toc93045598)

[4.2.2. Inițializarea modulului GPIO 8](#_Toc93045599)

[4.2.3. Inițializarea modulului PIT 10](#_Toc93045600)

[4.2.4. Inițializarea modulului ADC 13](#_Toc93045601)

[4.2.5. 15](#_Toc93045602)

[5. Dificultăți întâmpinate 17](#_Toc93045606)

[6. Referințe 19](#_Toc93045607)

# Prezentarea senzorului DFR0054

Senzorul DFR0054 este un senzor analogic, un simplu potențiometru (rotary potentiometer sensor) produs de DFRobot, parte a seriei Gravity, care permite rotația de la 0 până la 300 de grade.

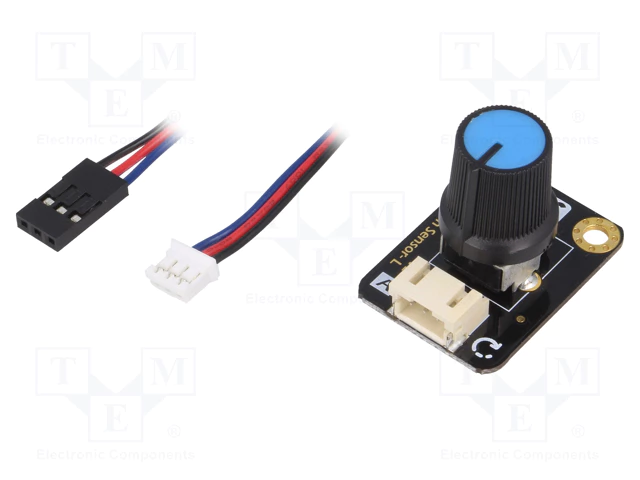


Figura 1. Senzorul DFR0045

**Descriere:**

* **Tipul Senzorului:** Potențiometru rotativ. Acesta este un senzor analogic care furnizează o valoare variabilă în funcție de poziția sau rotația unui potențiometru.
* **Interfață:** Analogică. Acest senzor se conectează la un port analogic al unui microcontroler sau unei plăci de dezvoltare pentru a transmite valori analogice.
* **Tensiune de Lucru:** Alimentare la 3.3VDC. Acesta necesită o alimentare de 3.3 volți pentru a funcționa corect.
* **Canale:** Are un singur canal (Ch: 1), ceea ce înseamnă că oferă o singură valoare de ieșire.

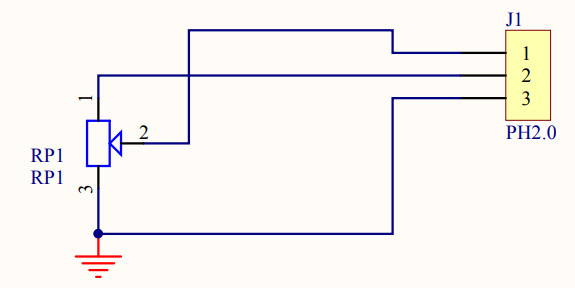


Figura 2. Circuitul senzorului DFR0045

Specificații:

* Tensiune de alimentare: 3.3V până la 5V;
* Se poate roti până la 300 de grade;
* Mărime: 22x31 mm.

|  |  |
| --- | --- |
| Red | VCC |
| Black | GND |
| Blue/Green | Analog signal output |

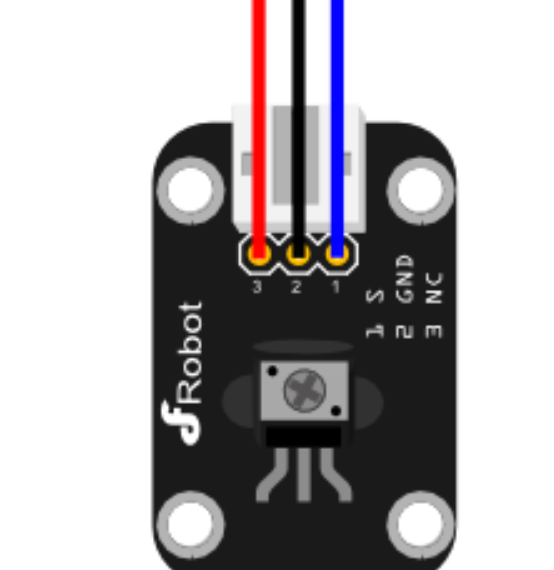


Figura 4. Conectare senzor

# Scop proiect

Scopul acestui proiect este acela de a afișa în interfață, un grafic ce reprezintă datele primite de la senzorul de roatie în timp real și modificarea LED-ulurilor, atat cele de pe breadboard, cat si cel prezent pe placa de dezvoltare pusă la dispoziție în funcție de poziția senzorului.

Se va dezvolta un program care, la un interval de 0.15 secunde, va verifica valorile primite de la senzor și va actualiza graficul.

De asemenea, sunt asignate patru culoari specifice, care trebuie sa fie afisate in ordine pe LED-ul dispus pe placuță, astfel: verde, albastru, mov si negru. Ordinea poate fi schimbata din interfata ( poate fi cea data anterior sau invers : nergu, mov, albastru si verde).

# Conectare senzor – placă de dezvoltare

Vom conecta senzorul astfel:

* Firul verde –
* Firul roșu –
* Firul negru – negru conectează senzorul la GND

Figura 5. Conectare senzori

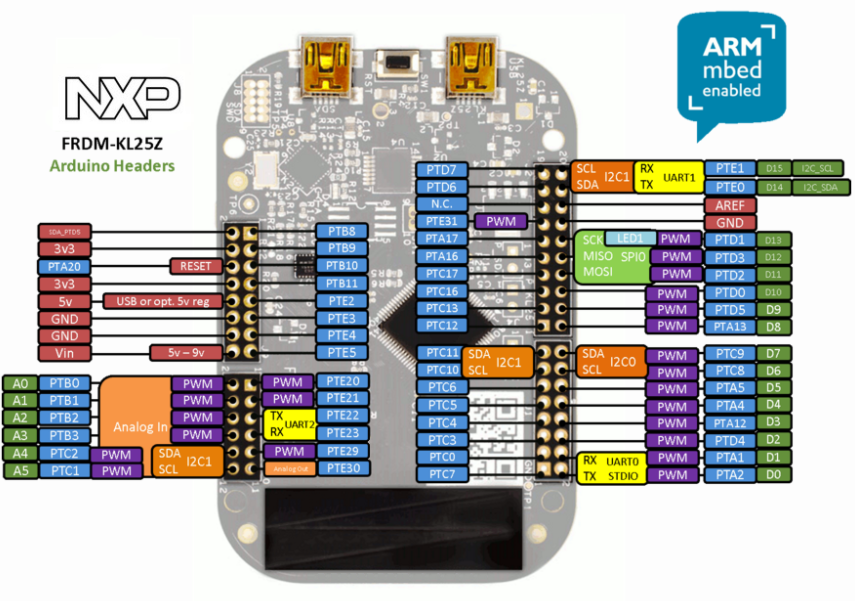


Figura 6. Pini placă dezvoltare[[1]](#footnote-1)

# Descriere program

## Funcția main

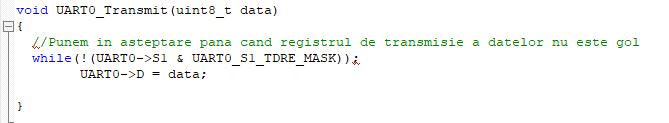
Figura 7. Funcția main

## Inițializarea modulelor

Figura . Funcția init

### Inițializarea modulului UART

Vom folosi modulul UART pentru comunicația serială si asincrona cu PC prin cablul USB. Are doua linii: receptie si transmisie.



return UART0->D : inseamna citire, iau de pe stiva.

### Inițializarea modulului GPIO

Modulul GPIO se ocupa cu pinii expusi fizic si permite controlarea la „run-time” a comportamentului unui pin al unui circuit integrat, mai specific, a direcției de trecere a curentului electric prin el (intrare / iesire) si valoarea logică pe care acesta o are.

Un pin poate fi configurat pentru scriere sau citire.

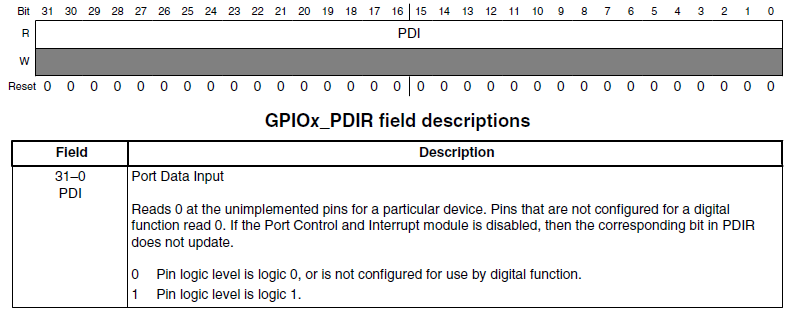
Interfațe (dispozitive externe) :

* Intrare – switch;
* Ieșire – LED;

Pe placa noi avem Port A, B, C si D. Fiecare port are 32 de biti (bit = pin fizic expus pe microcontroler), dar nu toti sunt expusi.

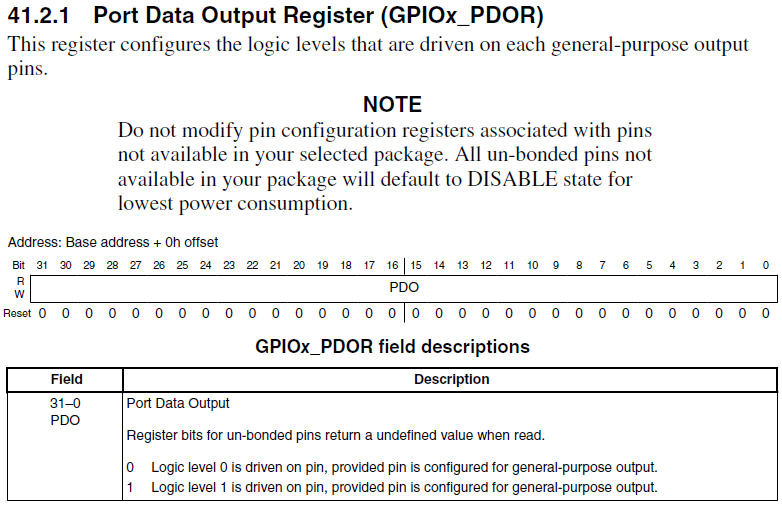
Pentru a controla un bit/pin, trebuie sa interactionam cu 3 registrii : PDIR, PDOR si PDDR:

* PDIR (GPIOx\_PDIR , unde x poate fi A, B, C sau D) : folosit pentru a mentine valoare bitului de intrare. Observam ca putem doar citi(R) din el, nu putem scrie(W).



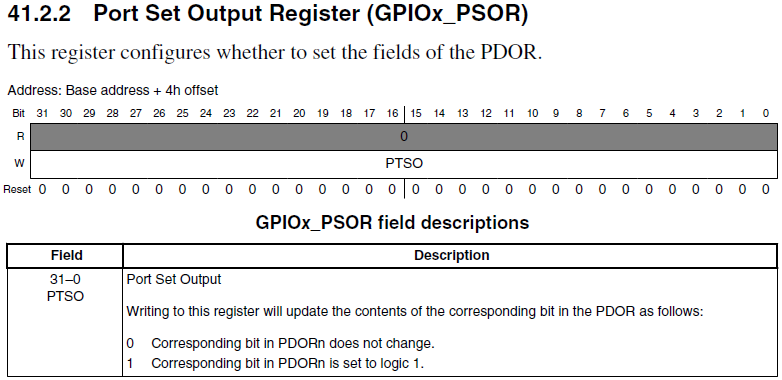
In cazul in care folosesc mai multi pini de pe acelasi port si acestia genereaza intreruperi, in handler-ul asociat pinului, cu ajutorul acestui registru GPIOx->PDIR pot sa vad verificand-ul, de pe care pin a venit intreruperea !!! (de exemplu folosim PTA4 si PTA12, pe ambii sunt conectate doua push-botton uri).

* PDOR (GPIOx\_PDOR , unde x poate fi A, B, C sau D) : este folosit pentru a scrie in el ceea ce vrem output.

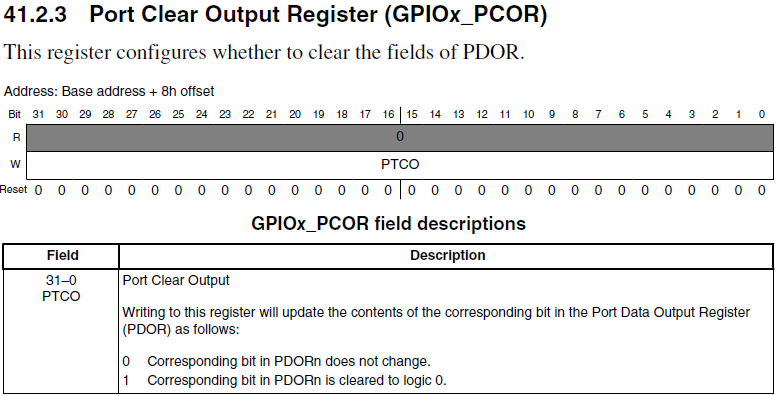


Putem modifica registru PDOR in 4 moduri :

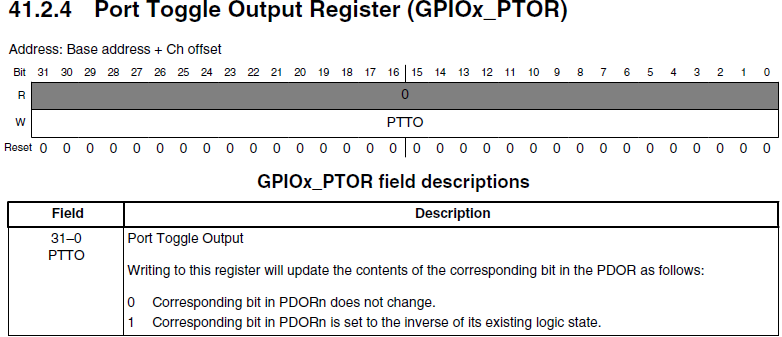
* Scriu direct in el
* Prin registrul PSOR:



* Prin registrul PCOR :

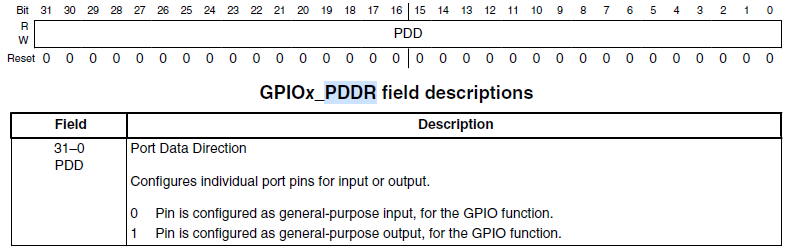


* Prin registrul PTOR:



* PDDR (GPIOx\_PDDR , unde x poate fi A, B, C sau D) : prin acest registru configurez/setez pinul ca fiind de input sau de output.

De exemplu, daca pentru pinul PTA4 registrul PDDR e setat pe 1, adica GPIOA->PDDR |= 1 << 4; atunci acel pin va fi folosit pentru output, iar daca e setat pe 0, acel pin va fi folosit pentru input.



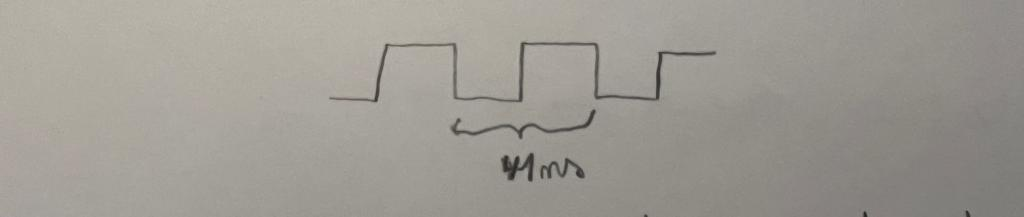
În continuare vom prezenta implementarea modului în proiectul nostru:

### Inițializarea modulului PIT

Perifericul PIT e o entitate separata de procesor si poate functiona in paralel chiar daca noi facem debug. Perifericul PIT nu interfatează cu pini externi. In general, pentru un numarator (PIT) avem :

* o frecventa de intrare (Fclk) = frecventa sursei de ceas
* si rezolutia = pe cati biti e acel numarator, adica pana la cat numara.

Pe aceasta placa FRDM-KL25Z avem doua surse de ceas (acele 2 cipuri cu suprafata metalica, argintie : y1 si y2). Exista doua numaratoare care sunt descrescatoare si sunt pe 32 biti => numere de la 0 pana la 2^32 – 1. Frecventa Fclk este de maxim 24 MHz = > Pclk = 1/(24\*10^5)s = 0,041 \* 10^6s = 41ns.



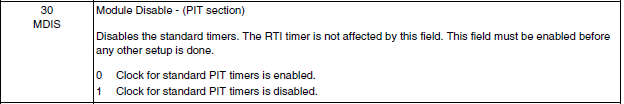
Pentru a mari rezolutia avem la dispozitie două canale care pot fi folosite pentru timere (Channel 0 si Channel 1, fiecare pe 32 biti). Daca le concatenam vom avea numarator pe 64biti = > numara de la 0 la 2^62 – 1 (se mareste plaja de valori). Accesul la un canal se face cu ajutorul sintaxei PIT->CHANNEL[x].

Fiecare timer are asociată o valoare de numărare independentă (load\_value = LDVAL). In cazul placii noastre, e o valoare de la xare incepe sa numere (sa decrementeze).

PIT ul, in general, are 3 registrii specifici:

* PIT\_MCR : This register enables or disables the PIT timer clocks and controls the timers when the PIT enters the Debug mode. Contine printre cei 32 de biti:

1. un bit pentru MDIS :

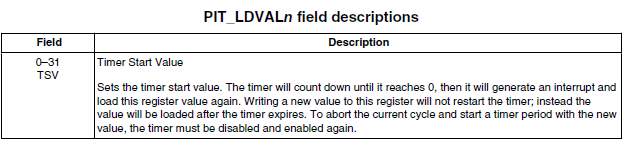


1. un bit pentru FRZ : daca e 0, timerul nu se blocheaza in cazul in care facem debug, iar daca e 1, atunci timerul este oprit.

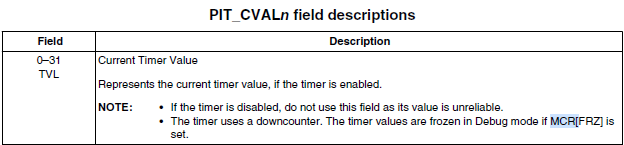
* PIT\_LTMR64H : acesta si urmatorul sunt pentru folositi pentru acea inlantuire/concatenare de timere
* PIT\_LTMR64L

Si fiecare timer/numarator/PIT la randul sau, are cate 4 registrii specifici:

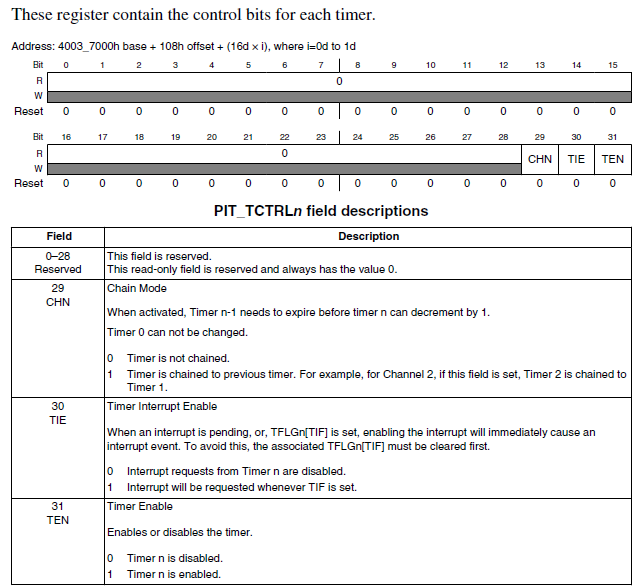
* PIT\_LDVALn (n e specific canalului, poate fi 0 sau 1) : reprezinta load\_value; prin el specific perioada dintre doua intreruperi(reseturi).



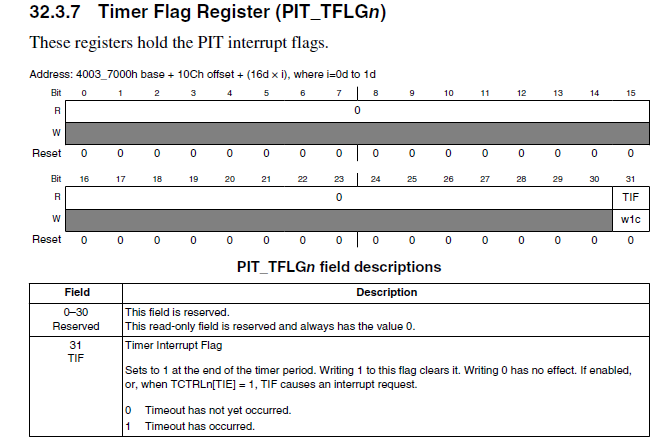
* PIT\_CVALn (n e specific canalului, poate fi 0 sau 1) : avem doar drept de read(R) si reprezinta valoarea curenta a timerului.



* PIT\_TCTRLn (n e specific canalului, poate fi 0 sau 1) :



* PIT\_TFLGn (n e specific canalului, poate fi 0 sau 1) :



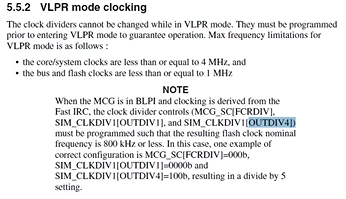
La proiectul nostru, modulul PIT este in stransa legatura cu led ul de pe placa, PIT fiind considerat device input iar ledul de pe placa, device output. Cu ajutorul lui vom afisa succesiunea de culori in oridinea : verde, albastru, mov si negru, sau invers.

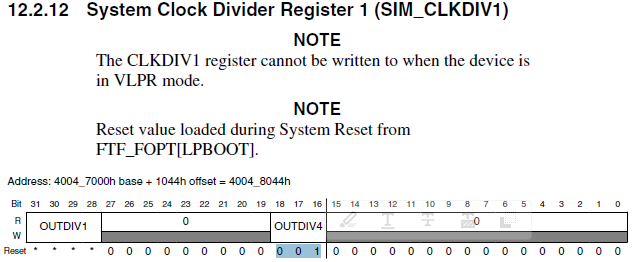
Orice periferic de pe aceasta placa va avea ca si frecventa valoarea lui Bus clock ( = frexcventa de ceas a magistralei). Pentru a calcula valoarea lui LDVAL avem nevoie sa stim valoarea lui bus clock care se calculeaza astfel (capitolul 5.4 din referance manual):



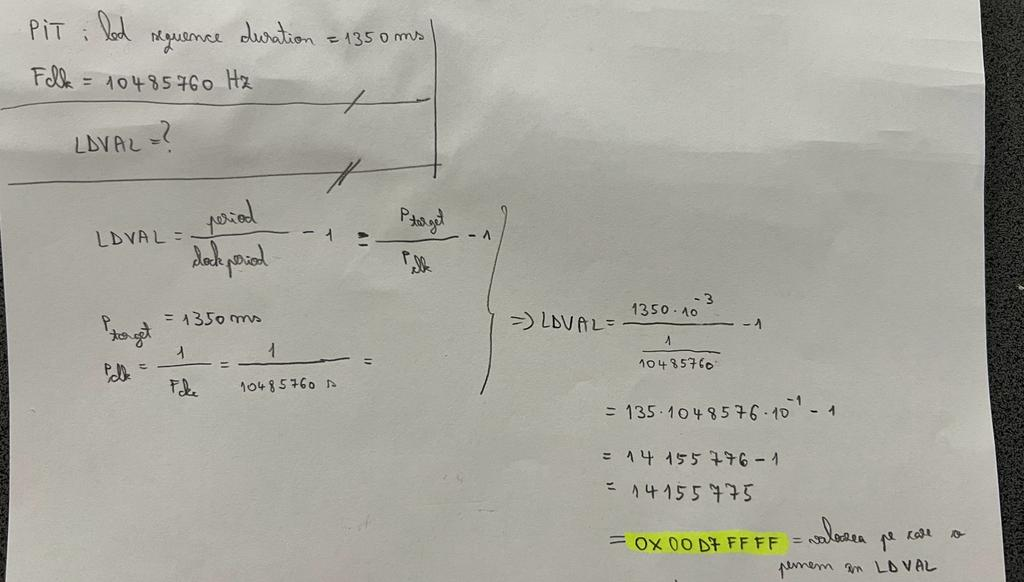
In  gasim valoarea lui system clock (e o valoare in hertz) :



Valoarea lui OUTDIV4 o vom lua cea de reset (vedem ca e 001) de unde e definita, si anume in registrul SIM\_CLKDIV : 



Ce inseamna 001? -> inseamna divide-by-2 = > ceea ce inseamna ca frecventa de intrare va fi DEFAULT\_SYSTEM\_CLOCK/2 = 20971520/2 = 10485760 Hz = frecventa maxima la care lucreaza microcontrolerul.



### Inițializarea modulului ADC

# Dificultăți întâmpinate

În timpul realizării acestui proiect, am întâmpinat mai multe dificultăți, precum:

* Nu stiam cum sa conectam ledurile pe breadboard. Rezolvare : Majoritatea becurilor LED au polaritate, ceea ce înseamnă că trebuie conectate corect pentru a funcționa. În general, pentru un LED:

**Anodul** este partea pozitivă și se conectează la sursa de tensiune sau la polul pozitiv al sursei de alimentare; poate fi identificat adesea prin sârmă mai lungă.

**Catodul** este partea negativă și se conectează la pământ (ground) sau la polul negativ al sursei de alimentare; este partea mai scurtă a sârmei.

# Referințe

[1] <https://www.seeedstudio.com/blog/2020/01/06/how-to-use-a-breadboard-wiring-circuit-and-arduino-interfacing/#:~:text=Plug%20the%20LED's%20longer%20lead,main%20part%20of%20the%20breadboard.&text=Plug%20one%20of%20the%20leads,middle%20channel%20of%20the%20breadboard>.

[2] FRDM-KL25Z\_ReferenceManual

[3] FRDM-KL25Z\_Schematics

[4] FRDM-KL25Z\_UserManual

|  |  |
| --- | --- |
|  |  |
|  |  |

1. [↑](#footnote-ref-1)